ESAME DI FONDAMENTI DI INFORMATICA T-2 del 13/9/2023
Proff. E. Denti – R. Calegari – A. Molesini
Tempo a disposizione: 3h30

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)
NOME CARTELLA PROGETTO: CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE: CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)
NOME JAR DA CONSEGNARE: CognomeNome-matricola.jar (es. RossiMario-0000123456.jar)

Si devono consegnare DUE FILE: *l'intero progetto Eclipse* e *il JAR eseguibile*

Si ricorda che compiti *non compilabili* o *palesemente lontani da 18/30* NON SARANNO CORRETTI e causeranno la verbalizzazione del giudizio "RESPINTO"

Nella ridente città di Dentinia è festa grande: ha appena aperto i battenti un nuovo mirabile fast food, DentBurger! L'applicazione deve quindi consentire agli utenti di ordinare il proprio pasto scegliendo dal vasto menù.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

I prodotti offerti da DentBurger si distinguono in diverse *categorie*: bevande, piatti, contorni, condimenti e dessert.

Per ogni categoria, il menù offre uno o più *generi*: ad esempio, come piatti può offrire panini, insalate, snack; come condimenti, patatine, chips, o altro; e così via.

Per ogni genere sono presenti nel menù uno o più *prodotti*, ognuno caratterizzato quindi da:

- categoria
- genere (denominazione)
- specifica dettagliata
- prezzo

Il cliente compone il proprio ordine scegliendo prodotti dal menù tramite la GUI: in ogni istante la GUI mostra la composizione dell'ordine e il prezzo complessivo, consentendo di aggiungere/rimuovere elementi.

Il file di testo DentBurgerList.txt, descritto più oltre, contiene i prodotti offerti a menù.

TEMPO STIMATO PER SVOLGERE L'INTERO COMPITO: 2h15 – 3h

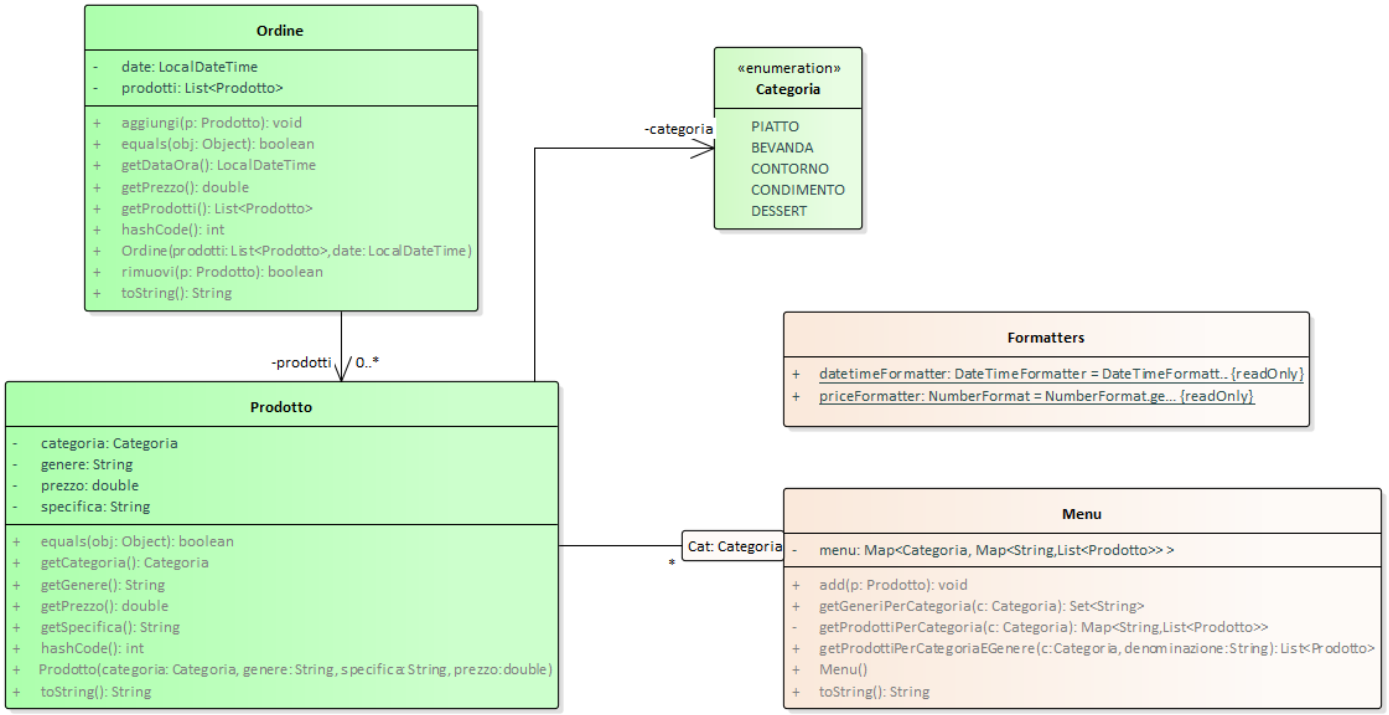
PARTE 1 – Modello dei dati: Punti 12 [TEMPO STIMATO: 55-70 minuti]
PARTE 2 – Persistenza: Punti 8 [TEMPO STIMATO: 40-50 minuti]
PARTE 3 – Grafica: Punti 10 [TEMPO STIMATO: 45-60 minuti]

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-19\lib"
--add-modules javafx.controls
```

Parte 1 – Modello dei dati (punti: 12)

Package: dentburger.model **[TEMPO STIMATO: 55-70 minuti]**



SEMANTICA:

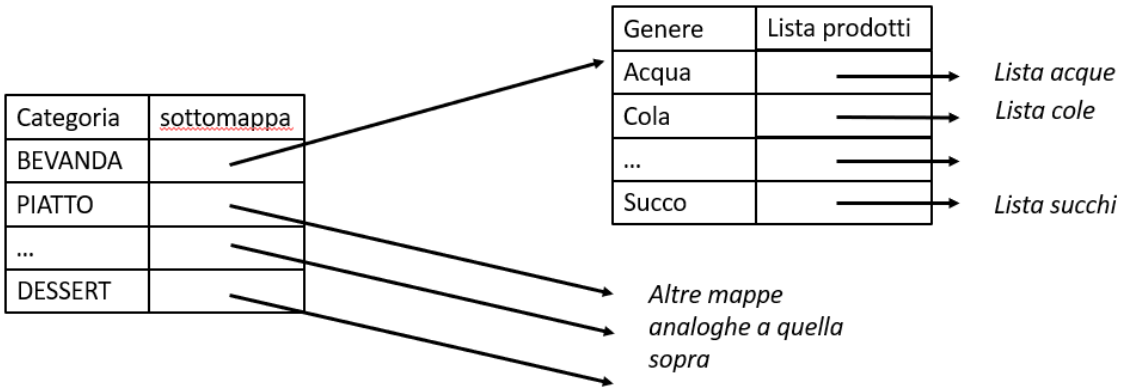
a) L'enumerativo ***Categoria*** (fornito) definisce le costanti corrispondenti alle categorie di prodotti previste

b) La classe ***Formatters*** (da completare) definisce i due formattatori personalizzati usati da tutta l'applicazione: in particolare, *datetimeFormatter* è un formattatore per data e ora secondo lo standard italiano SHORT, mentre *priceFormatter* è un formattatore per prezzi in Euro secondo lo standard italiano (quindi col simbolo di valuta posizionato dopo l'importo e da esso separato da spazio hard)

c) La classe ***Prodotto*** (fornita) rappresenta un singolo prodotto caratterizzato da ***Categoria***, genere (stringa), specifica (stringa) e prezzo. Il costruttore riceve e verifica gli argomenti in ingresso, lanciando apposite ***IllegalArgumentException*** con adeguato messaggio d'errore in caso di incoerenze. Appositi accessor consentono di recuperare le singole proprietà. Sono incluse opportune implementazioni di *equals*, *toString* ed *hashcode*. Da notare che, intenzionalmente, due prodotti sono considerati uguali se hanno le stesse proprietà a meno del prezzo: ciò al fine di prevenire che lo stesso prodotto possa comparire a menù più volte con prezzi diversi.

d) La classe ***Ordine*** (fornita) rappresenta un ordine inteso come lista di prodotti, effettuato in una certa data e ora. Il costruttore riceve e verifica gli argomenti in ingresso, lanciando due ***IllegalArgumentException*** distinte con adeguato messaggio d'errore in caso di argomenti nulli. Appositi accessor consentono di recuperare le singole proprietà. Sono incluse opportune implementazioni di *equals*, *toString* ed *hashcode*.

e) La classe ***Menu*** (da completare nell'implementazione) rappresenta il menù del ristorante, strutturato come mappa di mappe che gestiscono liste di prodotti. Più precisamente, il menù è organizzato come mappa indicizzata per ***Categoria***: a ogni categoria è associata una ulteriore mappa, indicizzata per genere (stringa), che associa a ogni genere la lista dei ***Prodotto*** di quella categoria e genere (vedere figura).



Il costruttore istanzia la mappa vuota: è compito del metodo *add* popolarla via via che vengono aggiunti prodotti, curando la costruzione delle mappe di secondo livello e delle liste quando necessario.

Devono essere implementati:

- Il metodo *add*, che aggiunge un nuovo prodotto nell'opportuna lista dell'opportuna mappa, se non già presente: se presente, deve lanciare ***IllegalArgumentException*** con apposito messaggio d'errore
- I due metodi *getGeneriPerCategoria* e *getProdottiPerCategoriaEGenere*, che restituiscono rispettivamente l'insieme (eventualmente vuoto) dei generi associati a una data categoria e la lista (eventualmente vuota) dei prodotti di una data categoria e genere.
  NB: a questo fine può essere utile sfruttare il metodo privato *getProdottiPerCategoria* (fornito), che restituisce la sottomappa relativa alla categoria fornita come argomento.

La classe definisce i vari accessor, nonché un'opportuna *toString*.

Parte 2 – Persistenza (punti: 8)

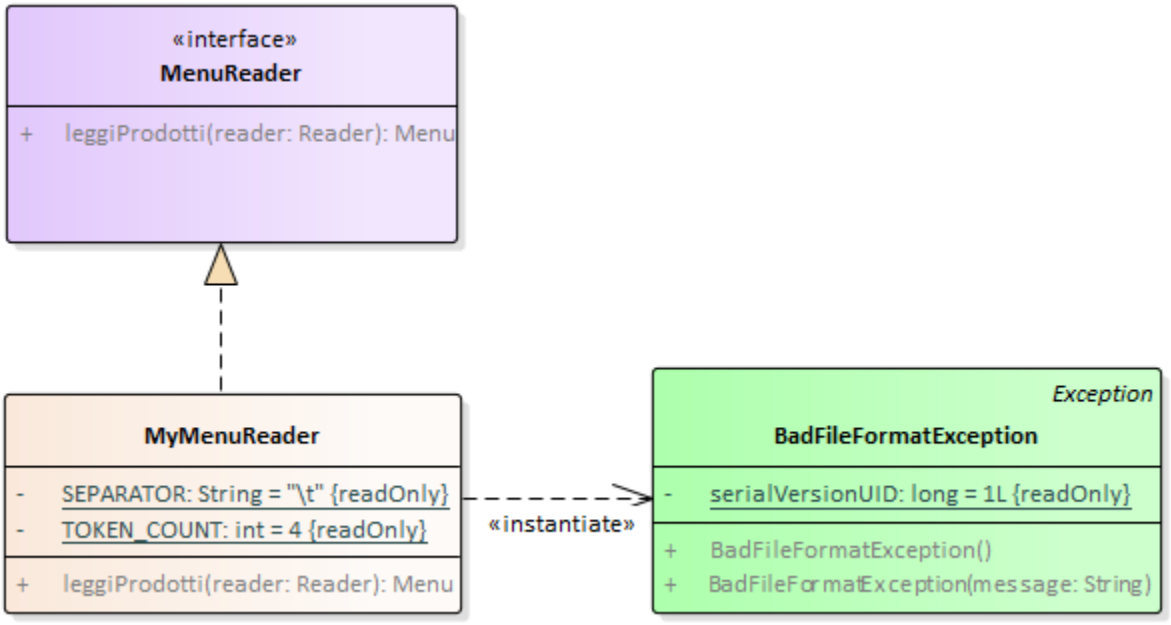
Package: dentburger.persistence **[TEMPO STIMATO: 40-50 minuti]**

Il file di testo DentBurgerList.txt contiene i prodotti offerti a menù dal ristorante, uno per riga. Ogni riga è composta di quattro elementi separati da una o più tabulazioni:

- Categoria del prodotto (stringa scritta con un qualunque insieme di caratteri maiuscoli e/o minuscoli)
- Genere del prodotto (stringa)
- Specifica del prodotto (stringa)
- Prezzo in Euro del prodotto, formattato secondo lo standard italiano col simbolo di valuta dietro al valore numerico, separato da uno spazio hard.

ESEMPIO

```
Piatto   Burger    Basic       4,90 €
Piatto   Pollo     Alette      3,90 €
Piatto   Insalata  Verde       3,50 €
Piatto   Insalata  Mista       3,80 €
Piatto   Insalata  con pollo   5,20 €
Contorno Patatine  Medie       2,70 €
Contorno Patatine  Grandi      3,60 €
Contorno Patatine  Formaggio   5,50 €
Bevanda  Cola      Media       3,50 €
Bevanda  Cola      Grande      4,50 €
Bevanda  Cola0     Media       3,80 €
Bevanda  Cola0     Grande      4,80 €
Bevanda  Acqua     Media       1,50 €
Bevanda  Aranciata Media       3,60 €
Dessert  Gelato    Panna       1,30 €
Dessert  Gelato    Panna e cioccolato  1,80 €
...
```



SEMANTICA:

a) L'interfaccia ***MenuReader*** (fornita) dichiara il metodo *leggiProdotti* che carica da un apposito Reader (già aperto) i dati necessari, restituendo un'istanza di ***Menu*** opportunamente popolata. Il metodo lancia:

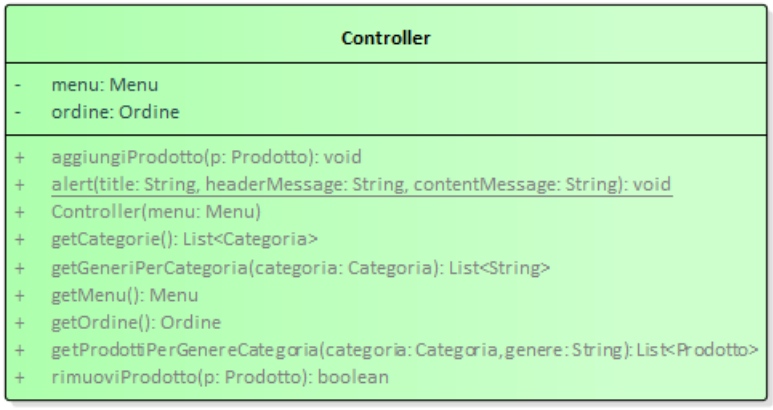
- ***IllegalArgumentException*** con opportuno messaggio d'errore in caso di argomento (reader) nullo;
- ***BadFileFormatException*** con messaggio d'errore appropriato in caso di problemi nel formato del file (mancanza/eccesso di elementi, categorie inesistenti, errori nel formato dei prezzi, etc.); in particolare, è richiesto di controllare con cura il formato del prezzo, verificando:
  a) che non siano presenti punti decimali
  b) che il numero sia letto interamente e non vi siano anomalie di sorta.
- una ***IOException*** in caso di altri problemi di I/O.

b) La classe ***MyMenuReader*** (**da realizzare**) implementa ***MenuReader*** secondo le specifiche sopra descritte.

Parte 3 (punti: 10)

Package: dentburger.controller *(punti 0)*

Il Controller è organizzato secondo il diagramma UML seguente:



SEMANTICA:

La classe ***Controller*** (fornita) riceve in fase di costruzione il ***Menu*** del ristorante, mantenuto nel proprio stato interno unitamente all'**Ordine** del cliente, che viene gestito interamente dal ***Controller*** stesso.

Sono forniti metodi per recuperare tutte le informazioni utili, molti dei quali delegano l'operazione al ***Menu*** o all'***Ordine*** interni. In particolare i tre metodi *getCategorie*, *getGeneriPerCategoria* *getProdottiPerGenereCategoria* restituiscono tre liste

utili per il successivo popolamento delle combo nella GUI, mentre i due metodi *aggiungi/rimuovi* consentono di modificare dinamicamente l'***Ordine*** del cliente aggiungendo/togliendo prodotti.

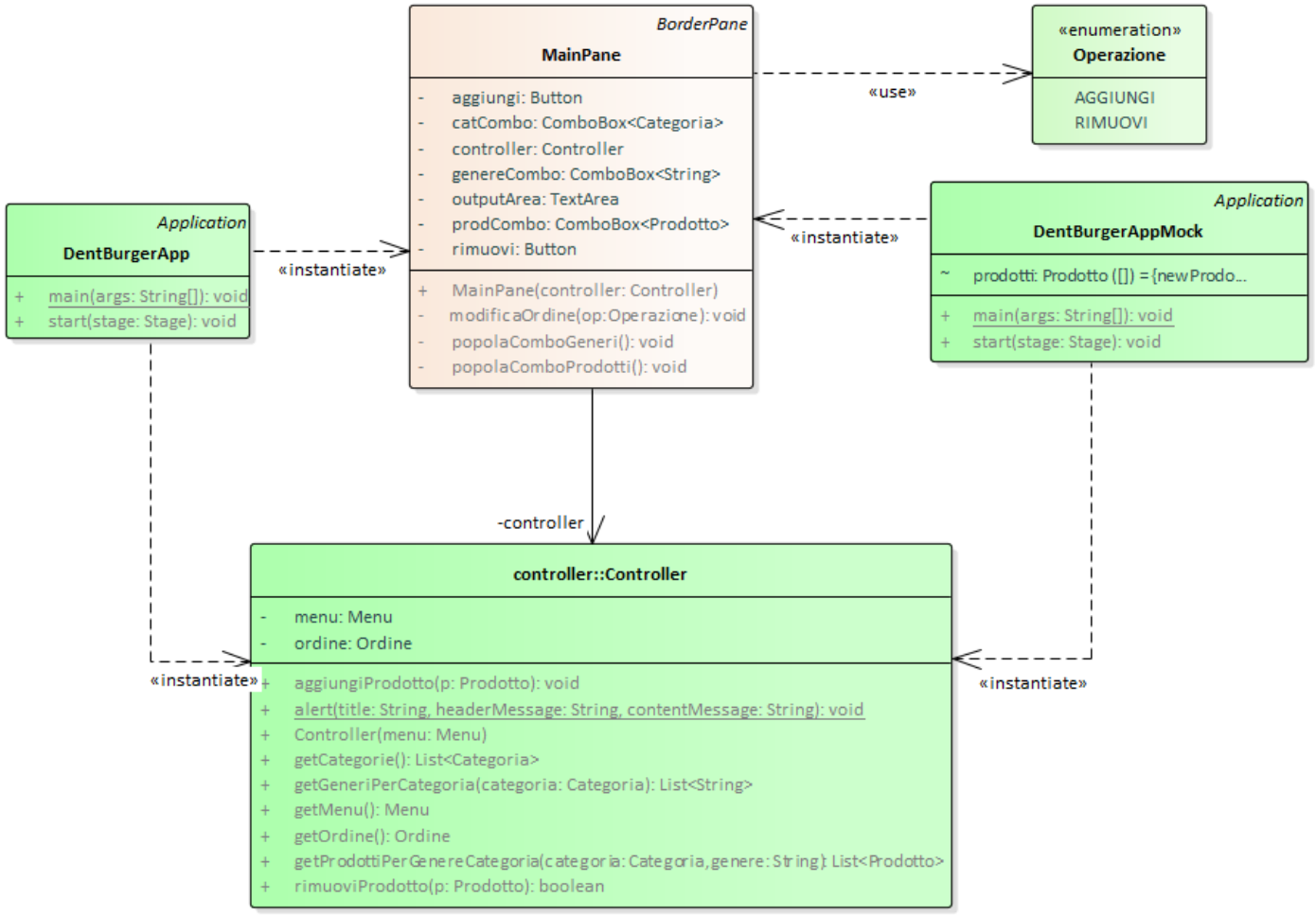
Infine, il metodo statico *alert*, utilizzabile anche dal MainPane, consente di far comparire all'utente, ove occorra, una finestra di dialogo con opportuno messaggio d'errore.

Package: dentburger.ui [TEMPO STIMATO: 45-60 minuti] *(punti 10)*

La classe ***DentBurgerApp*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire i file, creare il controller e incorporare il ***MainPane***. Per consentire di collaudare la GUI anche in assenza / in caso di malfunzionamento della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***DentBurgerAppMock***.

L'enumerativo ausiliario ***Operazione*** definisce le due costanti AGGIUNGI / RIMUOVI utili per parametrizzare l'ascoltatore degli eventi dei pulsanti.

L'interfaccia utente è illustrata nelle figure seguenti e segue il modello sotto illustrato*:*



L'interfaccia grafica si presenta come segue:

- a sinistra tre combo elencano rispettivamente le ***Categoria*** disponibili, i *generi* disponibili per la categoria selezionata e i prodotti disponibili per la categoria e il genere selezionati: pertanto, a parte la combo delle categorie che ha contenuto fisso, le altre due combo devono essere popolate /ripopolate dinamicamente, in base alle selezioni delle combo precedenti. Sotto, due pulsanti "Aggiungi" / "Rimuovi" consentono di aggiungere / togliere dall'ordine il prodotto attualmente selezionato.
- a destra, un'area di testo (inizialmente vuota e non scrivibile dall'utente) mostra in ogni momento lo stato dell'***Ordine*** dell'utente, unitamente al prezzo complessivo e alla data dell'ordine stesso (quella corrente), opportunamente formattati.

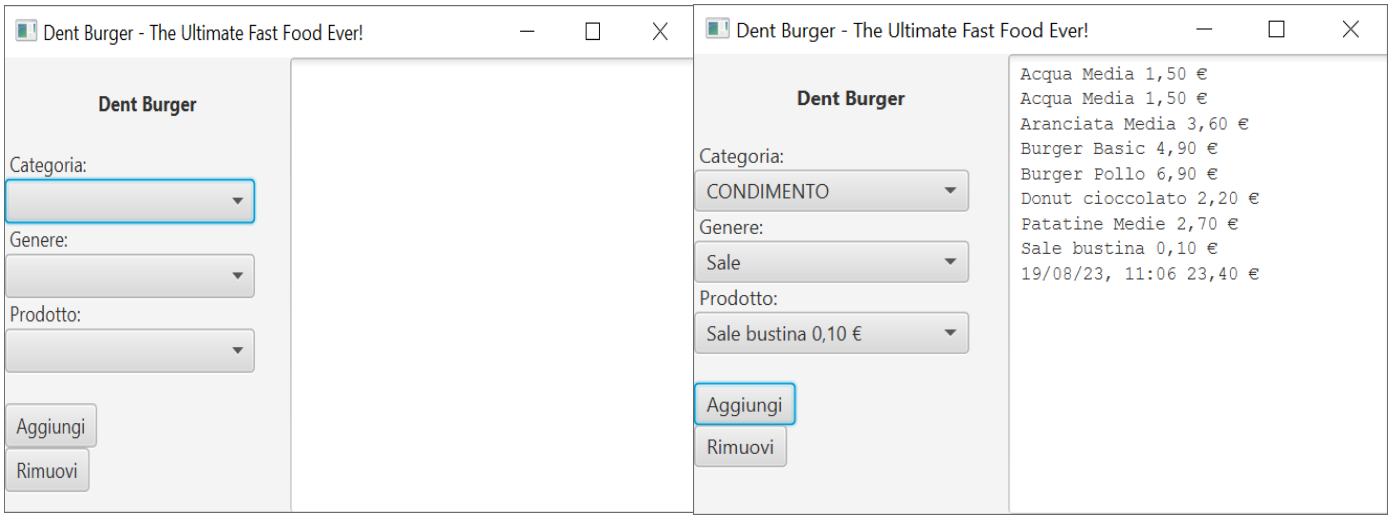


Fig. 1: a sinistra, la situazione iniziale della GUI, con tutte le combo ancora non selezionate, prima di qualunque interazione con l'utente; a destra, dopo la selezione e aggiunta di diversi prodotti.

Comportamento:

- la selezione di una ***Categoria*** dalla prima combo causa il popolamento automatico della seconda combo, quella dei generi, nonché la selezione automatica del primo elemento di essi; in cascata, ciò causa il popolamento della terza combo, quella dei prodotti, anche in questo caso con la selezione automatica del primo di essi.

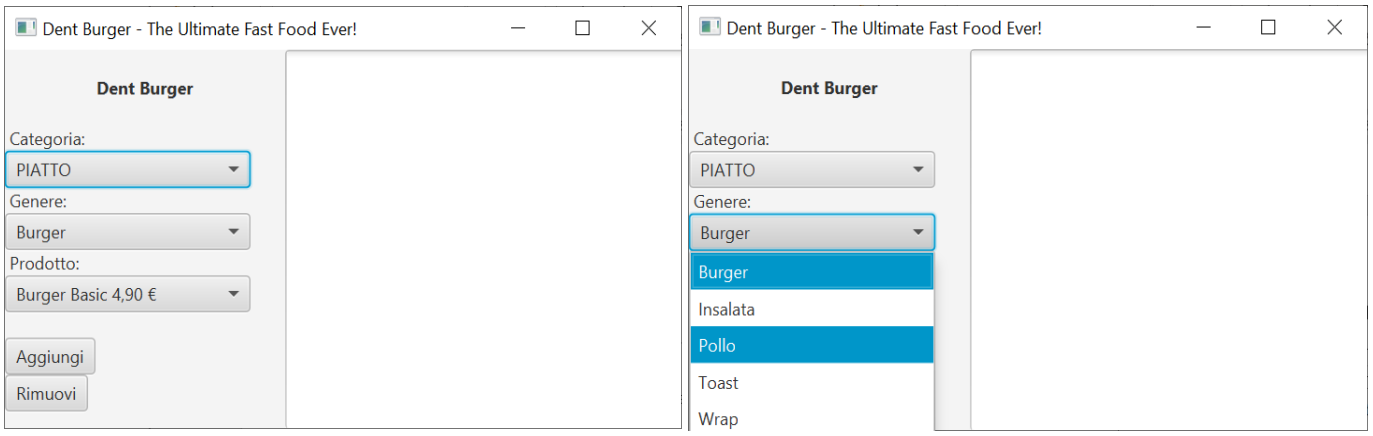


Fig. 2: a sinistra, la GUI dopo la selezione della categoria PIATTO dalla prima combo; come mostrato a destra, indipendentemente dalla selezione automatica è comunque possibile selezionare altri generi.

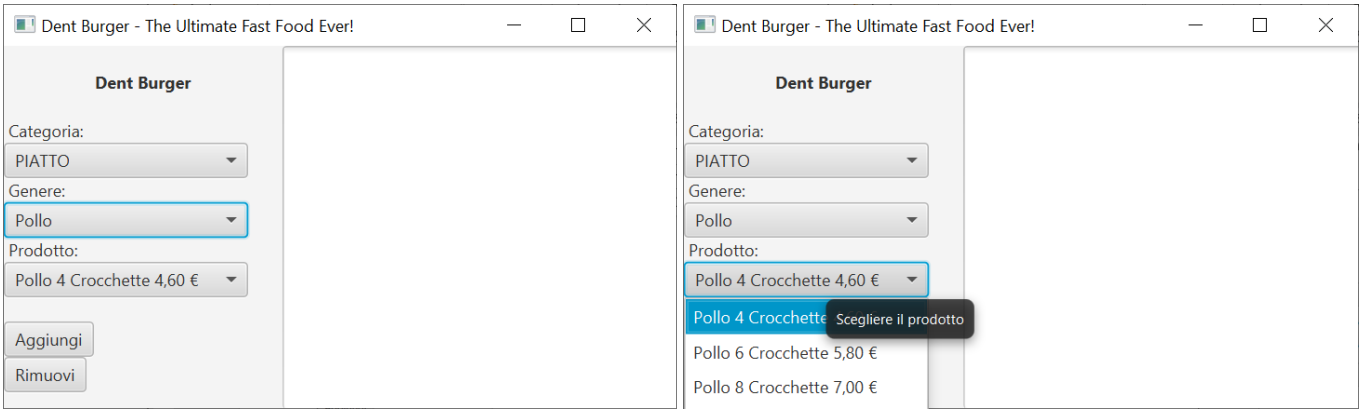


Fig. 3: a sinistra: se si seleziona un diverso genere (es. Pollo) la terza combo viene ripopolata di conseguenza con tutti i prodotti di quel genere (mostrati a destra).

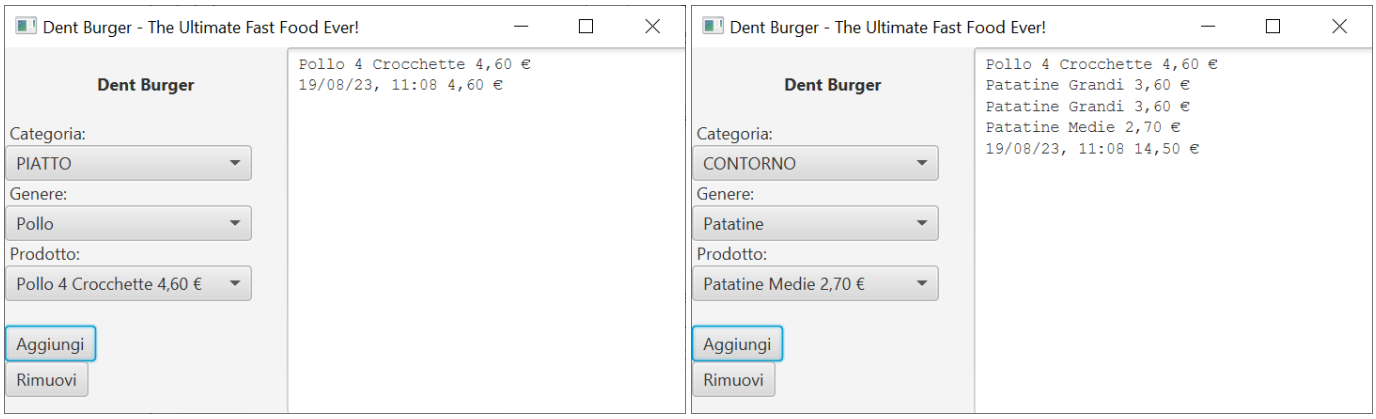


Fig. 4: a sinistra: premendo il pulsante Aggiungi, il prodotto selezionato viene incluso nell'ordine; procedendo in questo modo si possono aggiungere tutti i prodotti desiderati, anche più di una volta (a destra).

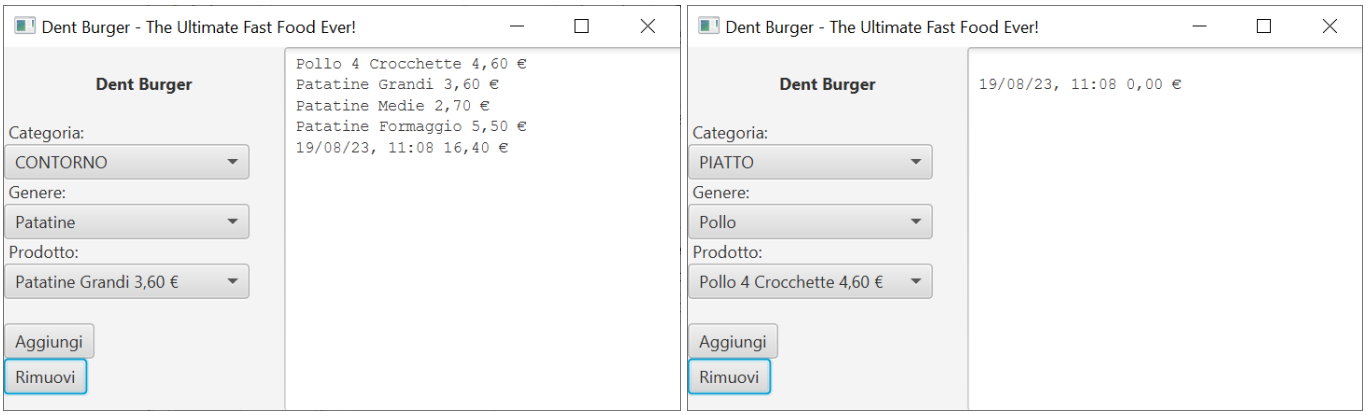


Fig. 5: a sinistra: premendo il pulsante Rimuovi è anche possibile togliere dall'ordine un prodotto precedentemente inserito (nell'esempio sopra, una delle due patatine grandi): nel caso esso non sia presente non accade nulla (non viene mostrato alcun messaggio d'errore). Da notare che il totale dell'ordine è sempre aggiornato dinamicamente. Come caso limite (a destra), se vengono rimossi tutti i prodotti, l'ordine si azzera.

Il MainPane è fornito *parzialmente realizzato*: è presente quasi tutta l'impostazione strutturale, mentre sono da completare la configurazione delle combo e la gestione degli eventi.

In particolare, ***MainPane*** estende ***BorderPane*** e prevede:

1) a sinistra, una ***VBox*** per le varie combo, i pulsanti e le etichette ausiliarie
2) a destra, una ***VBox*** con la sola area di output.

La **parte da completare** riguarda:

1) la configurazione iniziale delle tre combo
2) l'aggancio dei tre ascoltatori degli eventi, rispettivamente *popolaComboGeneri* per la combo delle categorie, *popolaComboProdotti* per la combo dei generi e *modificaOrdine* per i due pulsanti Aggiungi/Rimuovi.
3) la logica di gestione degli eventi, incapsulata nei tre metodi privati sopra citati.

La logica di gestione degli eventi è la seguente:

- *popolaComboGeneri* recupera dalla combo delle categorie la ***Categoria*** selezionata (se esiste, altrimenti non fa nulla) e, sulla base di quella, recupera dal ***Controller*** la lista dei soli generi di interesse: se tale lista ha almeno un elemento, provvede a selezionare il primo come default nella combo. Infine, richiama *popolaComboProdotti* per propagare il popolamento conseguente alla terza combo.
- *popolaComboProdotti* recupera dalle due combo delle categorie e dei generi la ***Categoria*** e il genere selezionati (se esistono: altrimenti non fa nulla) e, sulla base di questi, recupera dal ***Controller*** la lista dei soli prodotti di interesse: come sopra, se tale lista ha almeno un elemento provvede a selezionare il primo come default nella combo.

- *modificaOrdine* recupera dalla combo prodotti il ***Prodotto*** attualmente selezionato (se esiste: altrimenti emette un apposito messaggio d'errore tramite la funzione *alert* del ***Controller***), quindi distingue il pulsante premuto tramite l'argomento ***Operazione*** ricevuto e procede a invocare l'appropriato metodo del ***Controller*** per aggiornare l'ordine. Infine, aggiorna la visualizzazione dell'ordine – recuperato tramite l'apposito metodo del ***Controller*** – sull'area di output.

Cose da ricordare

- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

Checklist di consegna

- Hai fatto un **JAR eseguibile**, che contenga cioè l'indicazione del main?
- Hai controllato che **si compili** e **ci sia tutto**? [NB: non includere il PDF del testo]
- Hai **rinominato** IL PROGETTO, lo ZIP e il JAR esattamente come richiesto?
- Hai **chiamato** la cartella del progetto esattamente come richiesto?
- Hai fatto un **unico file ZIP (NON .7z, rar o altri formati)** contenente l'intero progetto? In particolare, ti sei assicurato di aver incluso tutti i file .java (e non solo i .class)?
- **Hai consegnato DUE file distinti, ossia lo ZIP col progetto e il JAR eseguibile?**
- Su EOL, hai **premuto** il tasto "CONFERMA" per inviare il tuo elaborato?